INFORME TP1

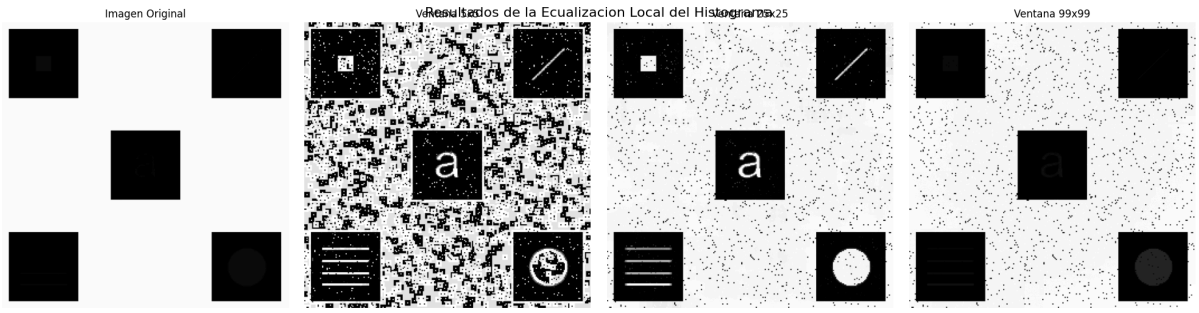
PROCESAMIENTO DE IMÁGENES, 2025

Bollini Lorenzo y Speranza Emanuel.

Problema uno - Ecualización local del histograma:

Para abordar el primer problema, se desarrolló una función denominada ecualizacion_local_del_histograma utilizando las bibliotecas numpy y cv2. Esta función implementa la técnica de ecualización local del histograma, la cual procesa una imagen en escala de grises deslizando una ventana de tamaño definible por el usuario a través de ella. Antes de iniciar el proceso iterativo, la función valida que las dimensiones de la ventana sean impares para garantizar un píxel central y aplica un borde a la imagen original mediante cv2.copyMakeBorder con cv2.BORDER_REPLICATE, replicando los valores de los píxeles del borde tal como se sugiere en la ayuda del enunciado .

El núcleo del algoritmo itera sobre cada píxel de la imagen original. En cada iteración, se extrae la ventana correspondiente de la imagen con borde y se aplica la función estándar cv2.equalizeHist a esta sub-imagen . El valor resultante del píxel central de la ventana ecualizada se asigna entonces a la posición correspondiente en una nueva imagen de salida. Finalmente, la función retorna esta imagen procesada. Para analizar los efectos de la técnica y la influencia del tamaño de la ventana, se cargó la imagen Imagen_con_detalles_escondidos.tif y se aplicó la función con ventanas de 5x5, 25x25 y 99x99. Los resultados fueron visualizados junto a la imagen original utilizando matplotlib.pyplot, permitiendo observar cómo diferentes tamaños de ventana afectan el realce de detalles locales. ¿La conclusión final? Utilizando 99x99 no se encontraron los detalles ocultos. Si se podían ver con 5x5, pero el ruido de la imagen hizo que nos decantamos por 25x25 como la mejor opción.



Original 5x5 25x25 99x99

Problema dos - Validación de formularios:

El segundo problema consistió en crear un script en Python para validar automáticamente formularios digitalizados, basándose en un conjunto de reglas específicas detalladas en el enunciado.

La aproximación principal se basa en la detección automática de la estructura de la tabla para localizar las celdas de interés, eliminando la necesidad de coordenadas fijas. Esto se logra a través de la función detectar_lineas, que primero aísla las líneas horizontales y verticales utilizando operaciones morfológicas (cv2.erode, cv2.dilate) para minimizar la interferencia del texto contenido en las celdas. Posteriormente, aplica la técnica sugerida en la AYUDA 1 : calcula las proyecciones de píxeles sumando los valores de la imagen de líneas por filas y columnas (np.sum) y detecta las posiciones de las líneas encontrando los picos en estas proyecciones mediante un umbral (UMBRAL_PICO). La función auxiliar encontrar_centros_lineas determina el centro preciso de cada línea detectada.

Una vez detectadas las coordenadas de las líneas (filas, cols) en el bucle principal que procesa cada formulario (formulario_*.png), se mapean estas coordenadas para definir dinámicamente las regiones de interés (ROIs) de cada campo (nombre, edad, etc.), aplicando un pequeño margen interno (MARGEN). A continuación, se extrae el contenido de cada ROI binarizada utilizando la función extraer_caracteres_y_palabras, la cual implementa la AYUDA 2 . Esta función emplea cv2.connectedComponentsWithStats para identificar componentes conectados (caracteres o marcas), los filtra por área mínima (MIN_CHAR_AREA) para descartar ruido, cuenta los caracteres (c) y estima el número de palabras (p) basándose en la distancia horizontal entre componentes, comparada con SPACE_THRESHOLD.

Los conteos (c, p) obtenidos para cada campo se pasan a la función validar_campos. Esta función aplica rigurosamente las reglas especificadas en

el Punto A del enunciado para determinar si cada campo es "OK" o "MAL". Los resultados de esta validación se imprimen directamente en la consola, cumpliendo con el requisito de salida del Punto A .

Finalmente, para cumplir con los Puntos C y D, los resultados de la validación de cada formulario se almacenan. Se genera un archivo reporte_formularios.csv que contiene el ID del formulario y el estado ("OK"/"MAL") de cada campo por fila . Simultáneamente, se guarda el recorte del campo 'nombre' de la imagen original en escala de grises. Una vez procesados todos los formularios, estos recortes se combinan en una única imagen reporte_crops.png, añadiendo un borde de color (verde para OK, rojo para MAL) a cada recorte como indicador visual, según lo solicitado en el Punto C .